Fitting Optimal Classification Trees Using an Elitist Genetic Algorithm with Bernoulli Crossover

Vanesa Salinas Comuzzi and Theodore T. Allen
Department of Integrated Systems Engineering, Ohio State University, 210 Baker Systems Building, 1971 Neil Avenue, Columbus, Ohio 43210, U.S.A.
salinascomuzzi.1@osu.edu, allen.515@osu.edu

Abstract

Optimal Classification Trees (OCTs) and Optimal Regression Trees (ORTs) promise to provide empirical modeling methods with unparalleled combinations of accuracy and interpretability. Yet, the computation times of the fitting procedures, which are currently based on integer programming solvers and branch and cut methods, tend to grow exponentially with the problem size. Therefore, problems involving over 3,000 datapoints may be practically out of reach. The purpose of this thesis is to explore the feasibility of Genetic Algorithms (GAs) to efficiently solve classification problems using variants of the well-known Iris Data classification problem. By studying this problem, it is verified that GAs can solve the problem optimally within a few seconds. Also, the GA time grows linearly with respect to the number of generations, the number in the population, and the number of runs. Therefore, exploring GAs to solve OCTs and ORTs is a promising topic for future research.

Keywords: Genetic Algorithms, Optimal Classification Trees, Computer Experiments

1. Introduction

Decision tree models offer relatively accurate, interpretable, and computationally efficient solutions compared with currently available alternative empirical modeling methods. The first introduction to decision trees was Classification and Regression Trees (CART) (Breiman, 1984). They are well-liked because of their interpretability and are used in over 37,000 transitions. Yet, the estimation method for CARTs is fundamentally greedy, meaning that the final fit model is not optimal for accuracy goals. Even the less greedy methods building on CART which "look ahead" by one level, are still suboptimal (Murthy, 1995). Random forest models can create multiple (each greedy) trees at once, but their results are difficult to comprehend, and therefore difficult to explain (Ho, 1998). Optimal Classification Trees (OCT/OCT-H) offer the improved accuracy by performing the estimation problem optimally (Bertsimas and Dunn, 2017), i.e., all the branching variables and values are determined simultaneously and optimal. For data with up to 3,000 points, accuracy can be as high as 96 percent. Still, the OCT formulation is an integer program and solution runtimes scale exponentially with the number of data points and depth of the tree. For more than 3,000 data points obtaining an optimal solution is often prohibitively difficult even with modern computers and branch and cut solvers.

Genetic algorithms were first proposed in Rechenberg (1964) and further developed in Holland (1975). De Jong (1975) proposed "elitist" genetic algorithms that copy a fraction of the solutions considered best from generation to generation. By copying many solutions instead of only the best solutions, the random error associated with simulation would be less likely to cause the best solution to be lost. Genetic algorithms are often more effective at solving large integer programs. Their runtime scales linearly as you add more data points, while still offering the same level of accuracy. By combining a Bernoulli crossover algorithm and a genetic algorithm, an efficient, accurate, and interpretable answer can be found for large datasets.

2. Iris Data Example

Fisher's Iris dataset contains 50 samples from three species of Iris (Fisher, 1936). The length and width of the sepals and petals, in centimeters, are given and can be used to differentiate between the different species.

The data can be differentiated using a classification tree. The tree begins at the top level with all the data. The data is the split into 2 categories based on one variable and one variable value,

creating 2 branches. The 2 branches then have the possibility of being split again based on two other variables and variable values. This process is inherently greedy because it splits on one level without any regard for future levels. This finds the local optimum at each level, but not the global optimum for the entire tree.

3. Problem Statement

Fisher's Iris dataset can be solved to optimality using a Genetic Algorithm with Bernoulli Crossover (Hadj-Alouane, 1997). We know the solution is optimal because the objective value agrees with published optimal values (Dunn and Bertsimas, 2017). The proposed method relaxes general constraints by adding a penalty function and employs reproduction, crossover, and mutation operations. OCT-H is a more advanced method that employs nonparallel hyperplanes with respect to the axis to further split the data.

3.1 Mathematical Notation & Formulation

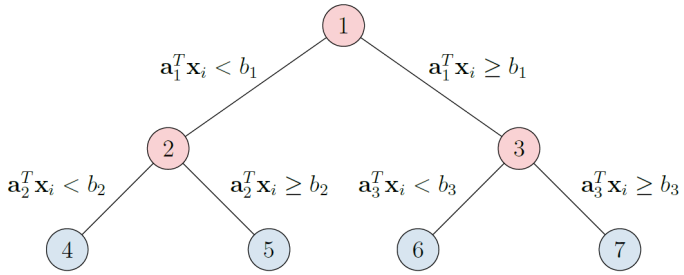

Figure 1: Maximal tree of depth 2.

Figure 1 shows how a datapoint would go through a classification tree to be added to a category. The problem constructs an optimal decision tree with a maximum depth of D and $T = 2^{D+1} - 1$ nodes, which are indexed by $t = 1, 2, \ldots, T$. The node notations of the model is as follows:

- $p(t)$ refers to the parent node of node t.
- $A(t)$ refers to the set of ancestors of node t. For example, node 5 has ancestors node 1 and 2. For example, $A(5) = \{1,2\}$ according to Figure 1.
- $A_L(t)$ refers to the set of ancestors of t whose left branch has been followed on the path from the root node to t. For example, $A_L(5) = \{1\}$ according to Figure 1.
- $A_R(t)$ refers to the set of ancestors of t whose right branch has been followed on the path from the root node to t. For example, $A_R(5) = \{2\}$ according to Figure 1.

- Branch nodes: $t \in \mathcal{T}_B = \{1, \dots, \lfloor T/2 \rfloor\}$ apply a split of the form $\boldsymbol{a}^T \boldsymbol{x} < b$ and $\boldsymbol{a}^T \boldsymbol{x} \geq b$. Points that satisfy this split follow the left branch in the tree, and those that do not satisfy it follow the right branch.
- Leaf nodes: $t \in \mathcal{T}_L = \{\lfloor T/2 \rfloor + 1, \dots, T\}$ make a class prediction for each point that falls into the leaf node.

The data is represented as $(\boldsymbol{X}, \boldsymbol{Y})$, containing n data points $(\boldsymbol{x}_i, y_i)$, $i = 1, \dots, n$. Each point has p features $\boldsymbol{x}_i \in \mathbb{R}^p$ and a class label $y_i \in \{1, \dots, K\}$. Therefore, the variables of the tree are defined as follows:

- $a_{jt} \in \{0,1\}$ denotes the split parameters (a.k.a, hyperplane coefficients). It indicates which feature $j \in \{1, \dots, p\}$ is selected to be split at branch node $t \in \mathcal{T}_B$.
- $b_t \in \mathbb{R}$ denotes the splitting threshold.
- $d_t \in \{0,1\}$ denotes the splitting indicator at branch node $t \in \mathcal{T}_B$.
- $z_{it} \in \{0,1\}$ indicates if data point $i \in \{1, \dots, n\}$ is assigned to leaf node $t \in \mathcal{T}_L$.
- $l_t \in \{0,1\}$ indicates if leaf node $t \in \mathcal{T}_L$ contains any point.
- $c_{kt} \in \{0,1\}$ indicates if class prediction $k \in \{1, \dots, K\}$ is assigned to leaf node $t \in \mathcal{T}_L$.
- $L_t \in \mathbb{R}^+$ refers to the misclassification error in leaf node $t \in \mathcal{T}_L$.
- $N_{kt} \in \mathbb{R}^+$ refers to the number of data points of class label $k \in \{1, \dots, K\}$ in leaf node $t \in \mathcal{T}_L$.
- $N_t \in \mathbb{R}^+$ refers to the total number of data points in leaf node $t \in \mathcal{T}_L$.

The determined parameters of the tree are defined as follows:

- x_{ij} denotes the feature $j \in \{1, \dots, p\}$ value for data point $i \in \{1, \dots, n\}$.
- y_i denotes the class label for data point $i \in \{1, \dots, n\}$.
- The objective is to minimize the misclassification error, so an incorrect label prediction has cost 1, and a correct label prediction has cost 0. Therefore, Y_{ik} is defined as follows:

$$Y_{ik} = \begin{cases} +1, & if\ y_i = k \\ -1, & otherwise \end{cases} \quad k = 1, \dots, K, \qquad i = 1, \dots, n$$

- $\hat{L}$ is the normalized term for misclassification error, and it could be calculated by simply predicting the most popular class for the entire dataset.

- α denotes the complexity parameter. If this parameter becomes larger, it will result in less splits.
- N_{min} denotes the minimum number of points in all leaf nodes.
- ϵ_j is used to prevent numerical instabilities in the MIO solver. To obtain this value, we sort the values of the *j*th feature and compute:

$$\epsilon_j = \min\left\{x_j^{(i+1)} - x_j^{(i)} \mid x_j^{(i+1)} \neq x_j^{(i)}, i = 1, \dots, n-1\right\}$$

  where $x_j^{(i)}$ is the *i*th largest value in the *j*th feature.
- ϵ_{max} is defined as the maximum value of ϵ_j for all features.

Finally, the formulation of the tree is given by:

$$\min \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t + \alpha \sum_{t \in \mathcal{T}_B} d_t$$

$$s.t.\ L_t \geq N_t - N_{kt} - n(1 - c_{kt}), \quad k = 1, \dots, K, \quad \forall t \in \mathcal{T}_L$$

$$L_t \leq N_t - N_{kt} + n \cdot c_{kt}, \quad k = 1, \dots, K, \quad \forall t \in \mathcal{T}_L$$

$$N_{kt} = \frac{1}{2} \sum_{i=1}^{n} (1 + Y_{ik}) z_{ik}, \quad k = 1, \dots, K, \quad \forall t \in \mathcal{T}_L$$

$$N_t = \sum_{i=1}^{n} z_{it}, \quad \forall t \in \mathcal{T}_L$$

$$\sum_{k=1}^{K} c_{kt} = l_t, \quad \forall t \in \mathcal{T}_L$$

$$\boldsymbol{a}_m^T \boldsymbol{x}_i \geq b_m - (1 - z_{it}), i = 1, \dots, n, \forall t \in \mathcal{T}_L, \forall m \in A_R(t)$$

$$\boldsymbol{a}_m^T (\boldsymbol{x}_i + \boldsymbol{\epsilon}) \leq b_m + (1 + \epsilon_{max})(1 - z_{it}), i = 1, \dots, n, \forall t \in \mathcal{T}_L, \forall m \in A_L(t) \quad (1)$$

$$\sum_{t \in \mathcal{T}_L} z_{it} = 1, \quad i = 1, \dots, n$$

$$z_{it} \leq l_t, \quad \forall t \in \mathcal{T}_L$$

$$\sum_{i=1}^{n} z_{it} \geq N_{min} l_t, \quad \forall t \in \mathcal{T}_L$$

$$\sum_{j=1}^{p} a_{jt} = d_t, \quad \forall t \in \mathcal{T}_B$$

$$0 \leq b_t \leq d_t, \quad \forall t \in \mathcal{T}_B$$

$$d_t \leq d_{p(t)}, \quad \forall t \in \mathcal{T}_B \backslash \{1\}$$

$$L_t \geq 0, \quad \forall t \in \mathcal{T}_L$$

$$z_{it}, l_t \in \{0,1\}, \quad i = 1, \dots, n, \forall t \in \mathcal{T}_L$$

$$a_{jt}, d_t \in \{0,1\}, \quad j = 1, \dots, p, \forall t \in \mathcal{T}_B$$

The explanation of the above formulation follows:

- The first and second constraints in Equation (1) represent the linearized form of misclassification loss in each leaf node, which is equal to the number of points in the node less the number of points of the most common label:

$$L_t = N_t - \max_{k=1,\dots,K} \{N_{kt}\} = \min_{k=1,\dots,K} \{N_t - N_{kt}\} \quad (2)$$

- The third constraint represents assigning data points of class label $k \in \{1, \dots, K\}$ to each leaf node $t \in \mathcal{T}_L$.
- The fourth constraint represents assigning all data points to each leaf node $t \in \mathcal{T}_L$.

- The fifth constraint enforces a single class prediction at each leaf node $t \in \mathcal{T}_L$ that contains points.
- The sixth and seventh constraints enforce the data points to follow the splits that are required by the structure of the tree.
- The eighth constraint ensures that each data point must be assigned to only one leaf node.
- The ninth and tenth constraints enforce a minimum number of data points at each leaf node.
- The eleventh, twelfth and thirteenth constraints enforce the tree structure.
- The last three constraints represent the feasible regions for those variables.

4. Genetic Algorithm Solution Method

The algorithm begins with the current data set, also known as the "population" or "current generation." Individual datapoints in the population are called chromosomes. The chromosome is typically stored in coded form, such as a vector of real numbers between 0 and 1. Each of these numbers is called a "gene."

The algorithm takes the highest *e* estimated fitness values, referred to as the elitist subset, and copies them into the next generation. In this case, e = 0.1 * the number of points in the population. The next subset of the population, m, is generated from pseudo-random numbers, or immigrants. In this case, the m = 0.1 * the number of points in the population. The final subset of the population is created using Bernoulli crossover. Two parents are randomly chosen, and two children are created from the parents by copying their alleles exactly. Then, a Bernoulli trial is run with p = 0.8. Each allele is tested and if the allele is above 0.8, the allele is flipped with the respective allele from the other child. Otherwise, the allele is left unchanged. This is done iteratively for the amount of specified generations. Then, the chromosomes are decoded, and the objective function is calculated.

4.1 Encoding Details

For the proposed method, the estimated fitness value *e* and the m value used is 0.1 multiplied by the number of datapoints in the population. Figure 2 below shows an example of a chromosome being decoded. Part (a) shows a matrix of 4 numbers that have been encoded into the (0,1) hyper-vector. Part (b) shows how the matrix was decoded into engineering units.

The algorithm finds the splitting variable by multiplying the allele by 4 and rounding. Then, the algorithm finds the splitting variable value by taking the allele and multiplying it by the maximum of the variable chosen to split on minus the minimum of the variable chosen to split on, and then adding the minimum of the variable chosen to split on. Finally, the branches in the tree are given classifications by multiplying the allele by 3 and rounding. Part (a) below shows the values of each number in chromosome. Part (b) shows an example chromosome. Part (c) shows a candidate solution. In this part, leaf 1 corresponds to the first species in the data set.

$$\begin{bmatrix} Branch\ variable \\ Variable\ value \\ Branch\ variable \\ Variable\ value \\ Branch\ variable \\ Variable\ value \\ Leaf\ 1 \\ Leaf\ 2 \\ Leaf\ 3 \\ Leaf\ 4 \end{bmatrix} \qquad \begin{bmatrix} 0.7 \\ 0.338 \\ 0.9 \\ 0.7 \\ 0.8 \\ 0.7 \\ 0.1 \\ 0.1 \\ 0.5 \\ 0.8 \end{bmatrix} \qquad \begin{bmatrix} ROUND(0.7 * 4) = 3 \\ 0.338 * (7.9 - 4.3) + 4.3 = 5.5168 \\ ROUND(0.9 * 4) = 4 \\ 0.7 * (7.9 - 4.3) + 4.3 = 6.82 \\ ROUND(0.8 * 4) = 4 \\ 0.7 * (7.9 - 4.3) + 4.3 = 6.82 \\ ROUND(0.1 * 3) = 1 \\ ROUND(0.1 * 3) = 1 \\ ROUND(0.5 * 3) = 2 \\ ROUND(0.8 * 3) = 3 \end{bmatrix}$$

(a) (b) (c)

5. Computational Results

The code was run for three different numbers of generations, population sizes, and number of data points. The code uses $\alpha = 0$ so that there is no penalty for including tree model nodes. Table 1 shows the result of the various runs. Note that the run time grows proportionally to the product of the number in the population times the number of generations. Table 2 shows the computation times for three different numbers of data points, i.e., the Iris Data is copied once and twice and appended. It is important to note that as the number of data points was increased, the runtime increased linearly. This shows the obvious potential advantage of genetic algorithms. While the computation effort for branch and cut may increase exponentially, the Genetic Algorithm may be able to achieve solutions with only a linear growth in computation times. Figure 2 below shows the trend in runtime from increasing the number of data points. The runtime scales nearly linearly.

Table 1: Accuracy and Runtime per Number of Generations and Population Size

	Number of Generations/Population		
	20/20	50/50	100/100
Fitness Accuracy (%)	96	96	96
Runtime (in secs)	5	40	170

Table 2: Accuracy and Runtime per Number of Data Points

	Number of Data Points		
	150	300	450
Fitness Accuracy (%)	96	96	96
Runtime (in secs)	5	13	21

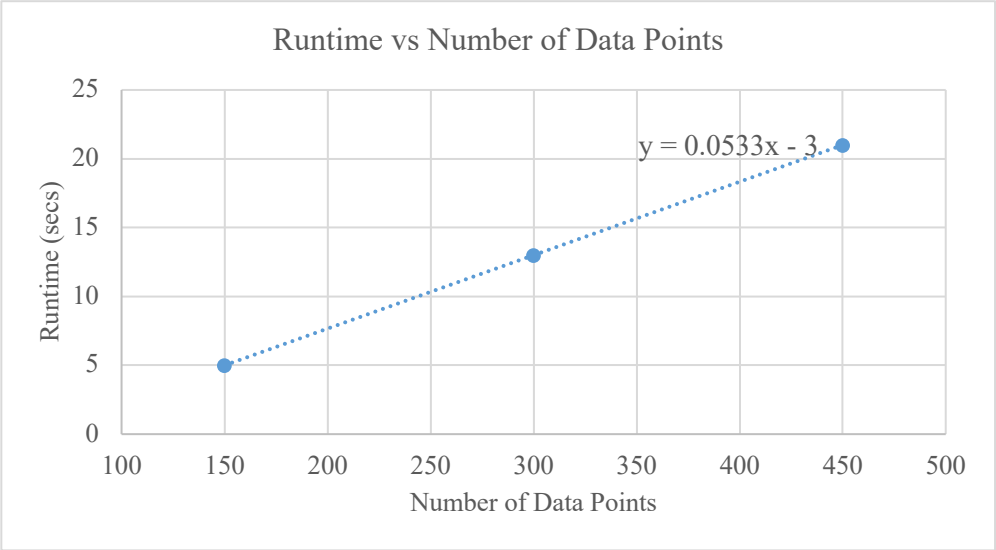

Figure 2: Runtime vs Number of Data Points

6. Conclusions

In this paper, we consider a classification problem using an optimal classification tree with Bernoulli crossover. This method is preferable to other similar methods of classification because of its efficiency, accuracy, and interpretability. CARTs are inherently greedy and random forests are difficult to explain. Decision tree models only offer up to one level of foresight and OCTs can only fit up to 3000 data points before losing accuracy. OCT with Bernoulli crossover allow for a

high degree of randomness, accuracy, and best of all, output a tree that is easy to see understand because each point can be easily followed on the tree branches.

Using Fisher's Iris dataset, we develop a mathematical model that codes each datapoint into a (0,1) hyperplane, categorizes each datapoint, and then decodes the point and calculates the accuracy of the prediction. We consider three different values for number of generations, population size, and number of data points. The runtime scales approximately linearly as the number of data points increases.

There are many opportunities for future research. First, the algorithm can be extended to include greater tree depths. The algorithm can also be optimized to reduce runtimes for adding more generations or increasing population size. Finally, the interpretability could be improved by adding a penalty for having additional nodes as Dunn and Bertsimas (2017) do in their formulation, i.e., solve with $\alpha \neq 0$.

7. References

Ben Hadj-Alouane, A., & Bean, J. C. (1997). A genetic algorithm for the multiple-choice integer program. Operations research, 45(1), 92-101.

Bernshteyn, M. (2001). Simulation optimization methods that combine multiple comparisons and genetic algorithms with applications in design for computer and supersaturated experiments (Doctoral dissertation, The Ohio State University).

Bertsimas, D. & Dunn, J. (2017). Optimal Classification Trees. Machine Learning, 106(7), 1039-1082.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and Regression Trees (Monterey, CA: Wadsworth and Brooks/Cole).

De Jong KA (1975) An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Ph.D. Dissertation. The University of Michigan, Ann Arbor

Dunn, J. W. (2018). Optimal trees for prediction and prescription (Doctoral dissertation, Massachusetts Institute of Technology).

Fisher, R. (1936). Iris flower dataset.

Holland, J. H. (1973). Genetic algorithms and the optimal allocations of trials. SIAM Journal of Computing, 2(2), 88-105.

Murthy, S., & Salzberg, S. (1995, August). Lookahead and pathology in decision tree induction. In IJCAI, 1025-1033.

Rechenberg I (1964) Kybernetische losungsansteuerung einer experimentellen forschungsaufgabe. Seminarvortrag, Hermann-Fottinger-Institut fur Stromungstechnik der Technische Universitat, Berlin